VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza videozáznamů ultrazvukových vyšetření

Analysis of Videorecordings of Ultrasonic Investigation

2011 Petr Zajíc

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2011 ……

Rád bych poděkoval doc. Ing. Lačezaru Ličevovi, CSc. za velmi trpělivé a ochotné vedení této práce a za jeho cenné rady a připomínky. Dále bych rád poděkoval své přítelkyni Ing. Janě Zamyslovské, která mi byla při psaní velikou oporou.

Abstrakt

Cílem této diplomové práce je vývoj modulu, který bude uživateli sloužit jednak jako jednoduchý přehrávač a navíc bude umět na základě událostí zachycených na EKG grafu uložit snímky, kterých se událost týká. Jelikož je EKG graf k dispozici pouze na videu, bude nutné provést analýzu tohoto videa a graf na něm správně rozpoznat. Vytvořený modul bude nutno začlenit do aplikace FOTOM 2009 a spolu s dalšími současně vyvíjenými moduly tak vytvořit novou verzi aplikace – FOTOM NG.

Klíčová slova: analýza obrazu, Java, fotogrammetrie, FOTOM 2008, FOTOM NG, NetBeans Platform, prahování, video, EKG

Abstract

The goal of this thesis is development of a Fotom module which is able to play video files and capture frames based on events. Since those events are related to the ECG shown on the video, some analysis will be needed to recognize the ECG. This module plugged together with two other simultaneously developed modules in the FOTOM 2009 will make a brand new version of this application – FOTOM NG.

Keywords: image analysis, Java, photogrammetry, FOTOM 2008, FOTOM NG, NetBeans Platform, thresholding, video, ECG

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
AWT	–	Abstract Window Toolkit
CCD	–	Charge-coupled Device
CMP	–	Cévní mozková příhoda
CT	–	Computer Tomography
ECG	–	Electrocardiograph
EDT	–	Event Dispatch Thread
EKG	–	Elektrokardiogramm
FIFO	–	First In, First Out
GAČR	–	Grantová agentura České republiky
GUI	–	Graphical User Interface
IDE	–	Integrated Development Environment
JAI	–	Java Advanced Imaging
JAR	–	Java Archive
JMF	–	Java Media Framework
JVM	–	Java Virtual Machine
MRI	–	Magnetic Resonance Imaging
px	–	Pixel
RGB	–	Red-Green-Blue

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Cévní mozková příhoda je třetí nejčastější příčinou úmrtí a nejčastější příčinou invalidizace v naší populaci. Díky tomu se stává významným problémem současnosti. Ischemické cévní příhody (iCMP) tvoří téměř 85% CMP. Jedno z řešení, jak snížit počet CMP v populaci je prevence a včasné odhalení ateromatózy krčních tepen. Duplexní ultrazvuková sonografie je v současné době nejvýhodnější metodou pro sledování progrese aterosklerotických plátů, především pro svou neinvazivnost, bezpečnost a dobrou reprodukovatelnost výsledků [14].

Cílem mé diplomové práce je vývoj modulu pro zpracování videozáznamů těchto ultrazvukových vyšetření. Modul bude poté integrován do systému FOTOM 2009, což je aktuální verze programů rodiny Fotom, a spolu s dalšími současně vyvíjenými moduly vytvoří novou verzi aplikace – FOTOM NG.

Diplomová práce je rozdělena do pěti kapitol, které se dále člení na podkapitoly. Kapitola následující za tímto úvodem je věnována popisu fotogrammetrie. Jedná se vědní obor sloužící v medicíně k vyšetření tkání neinvazivní metodou. Jako snímač se nejčastěji používá ultrazvuková sonda, která pracuje na principu odrazu ultrazvuku od tkání s různou hustotou.

Další kapitola pojednává o analýze medicínských snímků a čtenář se z ní mimo jiné dozví, jaké důvody mě vedly k vývoji modulu. V této části také uvádím, jak by se dal tento modul využít i v jiných odvětvích kromě lékařství.

Čtvrtá kapitola se zaměřuje na popis použitých softwarových prostředků. Zmiňuji se zde ve stručnosti o platformě NetBeans Platform a o vývojovém prostředí NetBeans IDE a také o přednostech a možnostech rozšíření Java Advanced Imaging. V této kapitole jsou rovněž nastíněny důvody, které mě vedly k výběru knihoven Xuggler. Ty slouží ke kódování a dekódování videa, s nímž se zabývám ve svém modulu.

V páté kapitole je vysvětleno, jaká funkčnost se od modulu vyžaduje a je popsán způsob, kterým jsem modul navrhnul. Zmiňuji se také o nejdůležitějších programátorských přístupech. Tato kapitola je rozdělena na dvě části. První část popisuje požadavky, návrh a implementaci video přehrávače a druhá je věnována analýze videa. Je zde také popsán algoritmus, který jsem použil při hledání důležitých míst v EKG grafu, který se na videozáznamu zobrazuje.

Na závěr jsou zhodnoceny dosažené výsledky a ověřena funkčnost vytvořeného modulu spolu s ostatními novými moduly, které vyvíjí Bc. Tomáš Pytlík a Tomáš Hudeček. Součástí vytvořeného modulu je uživatelská a programátorská dokumentace, která usnadní údržbu a vývoj dalších modulů.

2 Fotogrammetrie

2.1 O fotogrammetrii

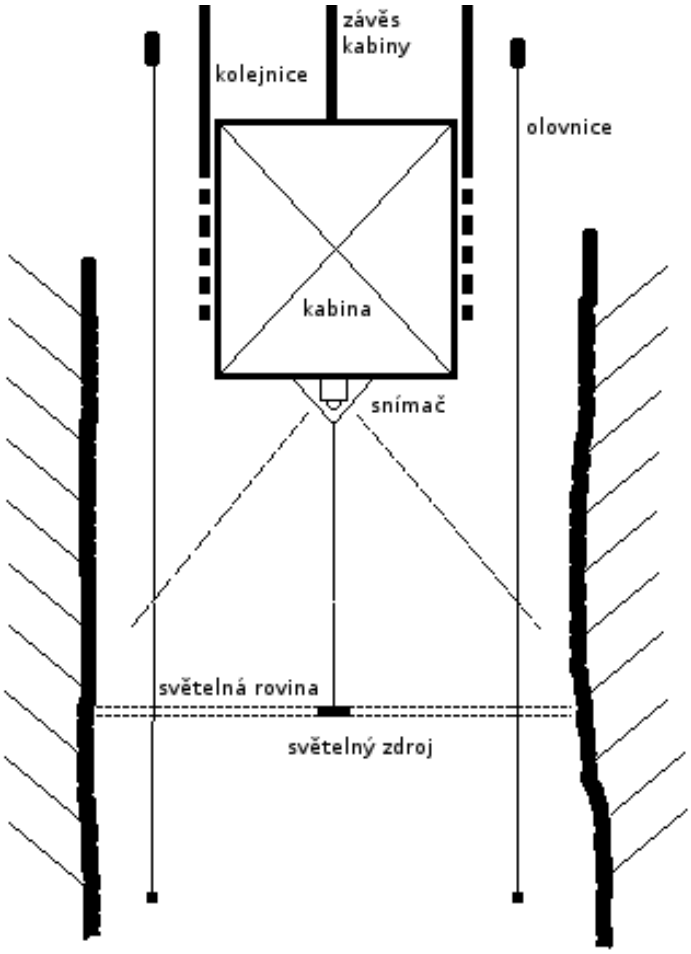
Fotogrammetrie je vědní obor, který se zabývá zpracováním informací a měřením na fotografických snímcích [3]. Fotografické snímky lze zkoumat díky speciálních přístrojů a systémů a změřené údaje poté zpracovat a vyhodnotit. Tuto disciplínu založil francouzský vědec Aimé Laussedat (*1819–†1907) a ve své práci *Métrophotographie* ji zmínil již v roce 1851. Fotogrammetrická metoda měření se zakládá na geometrických vztazích mezi předmětem a snímkem, které lze stanovit a poté použít k výpočtům a měřením. Pokud chceme měřit s velkou přesností, je nezbytné, aby se snímač vyznačoval dostatečnou rozlišovací schopností a citlivostí. Metody snímkování se nejprve zachytávaly na fotografický film na bázi bromidu stříbrného, ale později byl tento způsob nahrazen digitální fotografií a digitálním zpracováním. Měření se tedy neprovádí na fotografii, ale pomocí speciálního počítačového programu. K vytvoření digitálního snímku je potřeba skener, který digitalizuje standardně vyvolaný snímek anebo digitální kamera, ze které rovnou získáme digitální obraz. Digitální kamery pracují na principu polovodičového snímače CCD. Ten produkuje elektrický náboj dle intenzity dopadajícího světla. Elektrický náboj se poté vyjádří binárně [7].

Fotogrammetrii můžeme dle polohy snímače třídit na:

- **Leteckou** – v tomto případě se snímač nachází v letadle, vrtulníku nebo satelitu a vytvořené snímky jsou určené převážně pro tvorbu map.
- **Pozemní** – používá se k zachycení pohybu mostů, dokumentování historických budov nebo mapování hor.

2.1.1 Fotogrammetrie v hornictví

Za pozemní fotogrammetrii považujeme např. měření důlních šachet, u kterých je nezbytné monitorovat deformace profilu šachet, které s předstihem signalizují hrozící nebezpečí. Pomocí fotogrammetrie jsme schopni měřit bezpečně a pohodlně v jinak rizikových a těžce dostupných oblastech. K měření dochází ve svislých jámách tak, že kamera, nacházející se na těžní kleci, vytváří díky světelné rovině sérii snímků světelné stopy znázorněné na stěnách jámy (Obrázek 1). Pod klecí je umístěn speciální světelný zdroj, který tvoří světelnou rovinu. Blízko klece visí také olovnice, u kterých víme jejich vzdálenost a které se rovněž promítnou na světelné stopě, díky čemuž jsme schopni stanovit měřítko přepočtu mezi globálními (reálné vzdále-



Obrázek 1: Fotogrammetrie v hornictví

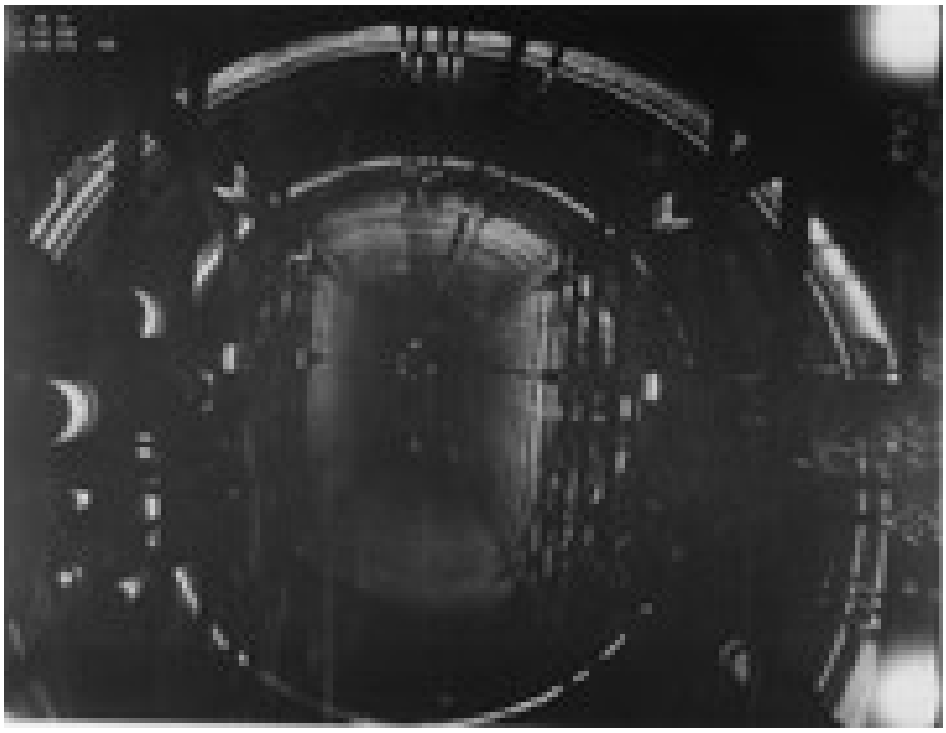
nosti) a lokálními souřadnicemi (zachycené na snímku). Při použití olovnic může vzniknout problém, který je zapříčiněn jejich rozkmitáním, což vede k nepřesnému výpočtu měřítka a tudíž i k nesprávnému měření. Obrázek 2 znázorňuje snímek pořízený jinou metodou bez aplikace zavěšených olovnic jako referenčních bodů. Využívá dvou rovnoběžných světelných rovin s pevně danou stálou vzdáleností [9].

2.1.2 Fotogrammetrie v medicíně

Fotogrammetrie v medicíně slouží k vyšetření tkání neinvazivní metodou (např. tepny, kosti, mozek). Jako snímač se obvykle využívá ultrazvuková sonda, která pracuje na principu odrazu ultrazvuku na rozhraní tkání s různou hustotou. Digitální obraz je možné spočítat pomocí času odrazu a síly ultrazvukového impulzu. Tento druh snímání je na rozdíl od CCD čipů značně náchylný k šumu, jelikož se mnohdy nemůže použít příliš velká intenzita ultrazvukového impulzu. Výpadky části obrazu představují další problém, který může nastat. Proto by měl fotogrammetrický program umět snímek nejen filtrovat a tlumit šum, ale i být schopen dopočítat chybějící informace na snímku, které jsou nutné k měření. Protože schopnost ultrazvuku pronikat kostmi není dostačující, slouží hlavně k vyšetření měkkých tkání.

Tvrdé tkáně (kosti) se zkoumají pomocí počítačové tomografie (CT). Tomografie je založena na rentgenovém záření, které proniká tělem a na druhé straně je s různou intenzitou zachytáváno detektorem. Obraz vzniká z velkého množství snímku dle změřené intenzity. Mezi značná pozitiva CT patří jeho obrovská přesnost a kvalita snímků.

Magnetická rezonance (MRI) skýtá další způsob vyšetření. Oproti CT se vyznačuje vysokým kontrastem mezi tkáněmi různé hustoty a tudíž je nejvhodnější pro onkologická a neurologická vyšetření (nádory, mozek). V těchto případech bychom pomocí tomografie ani ultrazvuku nezískali snímky v dostatečné kvalitě. Nevýhodou MRI je však vysoká cena a malá dostupnost, a proto nejsou vyšetření tímto zařízením příliš četná [7].



Obrázek 2: Snímání pomocí dvou světelných rovin

3 Analýza medicínských snímků za použití FOTOM NG

3.1 FOTOM NG

Již od roku 2001 je na katedře informatiky FEI VŠB-TUO vyvíjen systém FOTOM, který slouží k digitálnímu zpracování snímků [9][10]. Původně se tento systém používal pro zpracování důlních snímků a měření důlních jam, ale časem se z něj stal výkonný komplexní systém, který uměl detekovat různé zájmové objekty a vizualizovat měření. Jelikož byl původní program napsán jako jednoúčelový systém, nepočítalo se s žádnou možností jeho rozšíření a tak byl vývoj každého dalšího modulu stále složitější a díky absence pořádné dokumentace se z programu stávala velká „koule“ nesrozumitelného kódu.

Proto vznikl v roce 2009 úplně nový systém FOTOM 2009, jehož jádro bylo speciálně navrženo tak, aby podporovalo jednoduchý vývoj a integraci nových modulů. Společné API aplikace umožňuje komunikaci a spolupráci jednotlivých modulů, takže odpadá nutnost psát stovky řádků redundantního kódu. FOTOM 2009 byl napsán v programovacím jazyce Java a postaven na NetBeans platformě.

V současnosti probíhá vývoj nových modulů, které spolu s jádrem vytvoří zcela novou verzi aplikace FOTOM NG rozšířenou o možnosti 2D modelování, definici zájmových objektů pomocí Bézierovy křivky, přinese nové filtry pro zpracování obrazu a také analýzu různých videozáznamů.

3.2 Vyšetření krční tepny

Vyšetření krční tepny bylo řešeno v rámci grantových úkolů GAČR na 3D analýzu videa zachyceného ultrazvukovou sondou.

Důvodem ke snímání a následné analýze těchto medicínských snímků je snaha eliminovat vznik různých cévních onemocnění (především cévní mozkové příhody - CMP). CMP je třetí nejčastější příčinou úmrtí a nejčastější příčinou invalidizace v naší populaci. Díky tomu se stává významným problémem současnosti. Ischemické cévní příhody (iCMP) tvoří téměř 85% CMP. K nejvýznamnějším rizikovým faktorům iCMP patří hypertenze, srdeční arytmie, hypercholesterolémie a kouření.

Přes usilovné hledání a rozsáhlý výzkum dosud nebyla nalezena účinná terapie akutní iCMP, která by ve významné míře snížila úmrtnost a invaliditu. Proto je prevence jedním z řešení, jak snížit počet CMP v populaci. Prevence rizikových faktorů - ukončení kouření, dieta, terapie hypertenze, hypercholesterolémie a srdečních arytmií jsou možné cesty ke snížení vzniku CMP.

Nejčastější příčinou iCMP je ateromatóza krčních tepen. Podílí se na vzniku iCMP asi z 30%. Duplexní ultrazvuková sonografie je v současné době nejvýhodnější metodou pro sledování progrese aterosklerotických plátů, především pro svou neinvazivnost, bezpečnost a dobrou reprodukovatelnost výsledků [14].

Vzhledem k tomu, že se sonda snímající tepnu při každém úderu srdce pohne o 1 mm, je zapotřebí z ultrazvukového záznamu vytáhnout snímky, které zachycují stav tepny v určitém opakujícím se okamžiku. V každém intervalu (tj. doba mezi každými dvěma údery srdce) dochází totiž k roztáhnutí a opětovnému smrštění cévy. Cílem je zachytit cévu ve stejné míře roztaženosti jako v předchozím intervalu, i když intervaly v praxi nejsou stejně dlouhé. Je proto potřeba navrhnout a vytvořit modul pro automatické rozpoznávání těchto událostí, aby se co nejvíce minimalizoval lidský faktor.

3.3 Další možnosti použití

Modul pro analýzu videozáznamů ultrazvukových vyšetření nachází své využití i mimo medicínský sektor. Jeho nasazení je vhodné všude tam, kde vzniká video záznam sondou pohybující se konstantní rychlostí. Jako příklad uvedu měření výtahové šachty, štoly, tunelu. Nemusíme se ale omezovat jen na podzemní použití. Pod sondou si můžeme představit i letadlo, které letí konstantní rychlostí a snímá krajinu, která pod ním probíhá.

4 Softwarové prostředky

4.1 Netbeans Platform

Pro vývoj aplikací v Javě se nabízí několik různých vývojových prostředí. Výběr začíná u jednoduchých textových editorů, které umí jen zvýrazňovat syntaxi a končí u komplexních vývojových prostředí, které kromě zvýrazňování syntaxe nabízejí celou škálu dalších pomocných nástrojů (debugger, syntaktická analýza, zvýrazňování chyb v kódu, inteligentní doplňování kódu, profilování, GUI designér, a další), které mají programátorovi co nejvíce ulehčit práci. Mezi dvě nejznámější a nejrozšířenější patří NetBeans IDE a Eclipse. Obě tato vývojová prostředí nabízejí i své vlastní verze platforem. Co je to ale vlastně ta platforma?

NetBeans Platform je framework pro vývoj Swing aplikací. Jedná se o základní stavební kámen aplikace, který si dříve musel každý programátor napsat sám – ukládání stavu, vkládání akcí do menu, tvorba nástrojové lišty, stavový řádek, klávesové zkratky, rozvržení oken, aktualizace aplikace, atd. Vedle těchto základních schopností se vyznačuje NetBeans Platform i možností vyvíjet zásuvné moduly, které se mohou do hotové aplikace jednoduše nainstalovat.

Jelikož je systém Fotom 2009 postaven na architektuře NetBeans Platform, bylo celkem jasné, že svůj modul budu vyvíjet právě v prostředí NetBeans IDE. Diplomová práce pana Ing. Krahulce [7] mi poskytla slušný základ a zbytek informací byl dostupný v popisu NetBeans API [11].

4.2 Java Advanced Imaging

Kvůli omezeným možnostem standardního balíku Java2D bylo použito při vývoji jádra aplikace FOTOM 2009 rozšíření Java Advanced Imaging (JAI), které je vhodnější pro složitější transformace a zpracování složitějšími filtry. JAI se může pyšnit vysokým výkonem díky možnosti využití hardwarové akcelerace (například MMX u procesorů Intel), dále širokou škálou podporovaných formátů a v neposlední řadě i platformní nezávislostí [15]. Velké množství nástrojů, které JAI nabízí, bylo implementováno pro snadnější použití přímo do jádra aplikace FOTOM 2009.

4.3 Zpracování videa

Java nabízí pro práci s videem svůj vlastní framework JMF. Toto byla hned první možnost, kterou jsem mohl použít. Avšak po zjištění, že poslední verze tohoto frameworku (JMF 2.1.1e) byla vydána v roce 2003, jsem se rozhodl, že je tento framework již velice zastaralý a měl by problém zpracovat novější formáty videa. Začal jsem tedy pátrat po dalších možnostech.

Další alternativy JMF jsou třeba FMJ, který využijí hlavně vývojáři, kteří postavili svoji aplikaci na JMF, ale chtěli by přejít na něco novějšího. Jeho hlavní výhoda je v tom, že má stejné API, a proto je snadné vyměnit zastaralý JMF za FMJ. Dále stojí za zmínku Jffmpeg, DirectShow Java Wrapper, Fobs a nebo Xuggler. Při svém hledání jsem se setkal s různými diskuzními fóry, kde hodně uživatelů doporučovalo poslední zmiňovaný Xuggler, který byl proto další na řadě, abych ho vyzkoušel.

Již od začátku práce s Xugglerem mě velice zaujala pečlivě zpracovaná dokumentace [12] a kvalitní tutoriály, které by pochopil snad každý začátečník. Důležitá byla také možnost podpory od komunity uživatelů, které jsem mohl využít vždy, když se vyskytl nějaký problém. Xuggler také nabízí veškerou funkčnost, kterou jsem pro vývoj modulu potřeboval, a proto jsem se rozhodl, že pravě toto bude vhodný nástroj pro dekódování videa. V následující kapitole bych ho rád podrobněji popsal.

4.3.1 Xuggler

Xuggler je zapouzdření projektu FFmpeg, který umožňuje nahrávání, konverzi a streamování digitálního zvuku (audia) a obrazu (videa). Jelikož je použití projektu FFmpeg v jazyca Java nepohodlné a složité, vznikl kvůli tomuto účelu projekt Xuggler, který svým uživatelům nabízí multiplexování a demultiplexování všech nejpoužívanějších formátů. Pomocí jednoduchého příkazu si můžeme nechat vypsat na obrazovku všechny podporované formáty, kodeky a celkové nastavení Xuggleru (Výpis 1).

```
    java -cp %XUGGLE_HOME%\share\java\jars\xuggle-xuggler.jar
com.xuggle.xuggler.Configuration
```

Výpis 1: Ukázka výpisu všech možných nastavení Xuggleru

Za zmínku stojí všem známé formáty avi, flv, mov, mp4, 3gp, h264 a mnoho dalších.

Xuggler se skládá ze dvou komponent. Jednou z nich je sada JAR souborů a druhou je sada sdílených knihoven (.dll pro systém Windows, .so pro Linux a .dvlib pro Mac). Xuggler dále nabízí dvě základní API.

- ***MediaTool API***, které poskytuje jednoduché rozhraní pro rychlé psaní kódu
- ***Advanced API***, které nabízí programátorovi maximum z možností Xuggleru

MediaTool API Nyní bych rád ukázal, jak lze jednoduše pracovat s MediaTools API. Jak už jsem zmínil dříve, MediaTool je zjednodušené rozhraní pro práci se zvukem a videem. Schová-

vá všechny pokročilé detaily a nabízí přátelské prostředí pro rychlé psaní programů. MediaTool je založeno na systému posluchačů událostí (Výpis 2). Nejprve je vytvořen reader, který čte data ze zadaného souboru. Následně je k tomuto readeru přidán posluchač (writer), který obdrží všechna data, která reader přečte. Takto můžeme na pár řádků napsat velice jednoduchý program, který umí převést video z jednoho formátu do druhého.

```
IMediaReader reader = ToolFactory.makeReader("videofile.flv");
reader.addListener(ToolFactory.makeWriter("output.mov", reader));

while (reader.readPacket() == null) {}
```

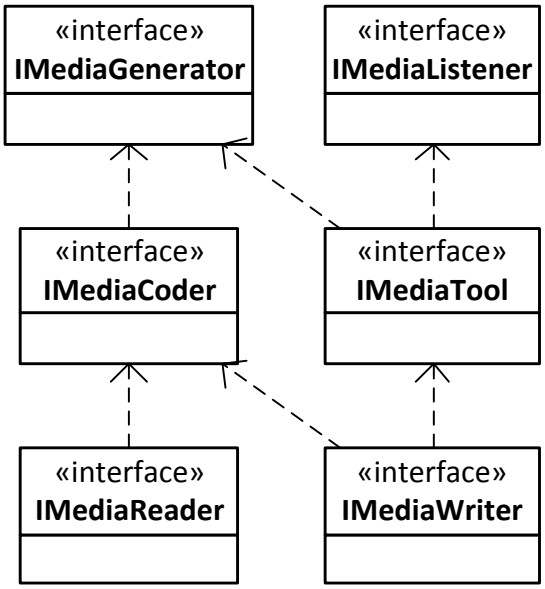
Výpis 2: Příklad použití MediaTool API

Jednotlivé posluchače je možné také řetězit za sebe do řady a do vstupního videa tak například vložit časovou známku, snížit hlasitost a výsledek převést do jiného formátu. Krátký příklad můžete vidět na následujícím výpise (Výpis 3). Opět se vytvoří reader jako v předchozím příkladu. Dále se také vytvoří writer a instance tříd, které se budou starat o vykreslení časové známky a snížení hlasitosti. Poté se nastaví samotné zřetězení, které můžeme zjednodušeně vyjádřit takto: reader → addTimeStamp → reduceVolume → writer.

```
IMediaReader reader = ToolFactory.makeReader("input.flv");
reader.setBufferedImageTypeToGenerate(BufferedImage.TYPE_3BYTE_BGR);

IMediaWriter writer = ToolFactory.makeWriter("output.mov", reader);
IMediaTool addTimeStamp = new TimeStampTool();
IMediaTool reduceVolume = new VolumeAdjustTool(0.1);

reader.addListener(addTimeStamp);
addTimeStamp.addListener(reduceVolume);
reduceVolume.addListener(writer);

while (reader.readPacket() == null){}
```

Výpis 3: Příklad řetězení posluchačů

Pro náročnější vývojáře je přichystáno komplexnější Advanced API s rozšířenými možnostmi. Programátor tak získá plnou kontrolu nad vytvářenými objekty a dostane se mu do rukou možnost nastavit si vše podle svých představ, např. nastavení kontejneru, streamu, paketů, atd. V modulu jsem použil Advanced API právě proto, že nabízí tyto rozšířené možnosti.



Obrázek 3: Zjednodušený třídní diagram MediaTool API

Advanced API Abychom mohli dekódovat video pomocí Advanced API, musíme si nejprve vysvětlit pár základních pojmů. Soubor s videem se označuje jako kontejner a většinou se skládá z hlavičky, paketů dat z různých stop a patičky. Hlavička obsahuje základní informace o tom, kolik stop kontejner obsahuje a informace o nich a také jaké kodeky byly použity na kompresi dat. Např. ve flv souboru zabírá tato hlavička prvních 9 bajtů. Dále následuje paket, který může obsahovat zvukovou stopu nebo obrazovou stopu a také časovou známku, která nám řekne, kdy by se měl paket přehrát, náleží-li ke zvukové stopě nebo kdy by se měl zobrazit, pokud se jedná o video stopu. Tento čas je v jednotkách, které si určuje kontejner. Pro formát flv to jsou např. milisekundy (1/1000 sekundy). Některé formáty podporují dokonce ještě další typ stopy, která obsahuje titulky. Samotná data jsou také většinou komprimována nebo zakódována, aby zabíraly méně místa. Na konci souboru zpravidla najdeme i patičku. Jelikož se v této práci zabývám pouze analýzou videa, zvukovou stopu při procházení paketů ignoruji.

V samotné implementaci je třeba nejprve otevřít kontejner neboli soubor s videem. Kontejner reprezentuje objekt typu *IContainer*. Dále se kontejneru zeptáme, kolik stop obsahuje a rovnou si je všechny projdeme, abychom zjistili, o jakou stopu se jedná. Jak už jsem zmínil, zvukovou stopu můžeme ignorovat, takže stačí zjistit, v jaké stopě se nachází obraz. Jednotlivou stopu reprezentuje objekt typu *IStream*. Jakmile najdeme obrazovou stopu, začneme procházet všechny pakety a z těchto paketů složíme jednotlivé snímky videa, které se ve správný čas zobrazí na obrazovku. Jednoduchou ukázku kódu pro dekódování videa nabízí Výpis 4.

```
IContainer container = IContainer.make();
container.open(filename, IContainer.Type.READ, null);
int numStreams = container.getNumStreams();
int videoStreamId = -1;
IStreamCoder videoCoder = null;
for (int i = 0; i < numStreams; i++) {
    IStream stream = container.getStream(i);
    IStreamCoder coder = stream.getStreamCoder();

    if (coder.getCodecType() == ICodec.Type.CODEC_TYPE_VIDEO) {
        videoStreamId = i;
        videoCoder = coder;
        break;
    }
}
IPacket packet = IPacket.make();
while (container.readNextPacket(packet) >= 0) {
    // Dekodovani paketu a sestaveni obrazu
}
```

Výpis 4: Ukázka dekódování videa pomocí Advanced API

5 Návrh a implementace modulu

Hlavním úkolem bylo vytvořit modul, který půjde lehce začlenit do stávající aplikace FOTOM 2009 a který bude maximálně využívat možnosti jeho jádra.

5.1 Video přehrávač

5.1.1 Specifikace požadavků

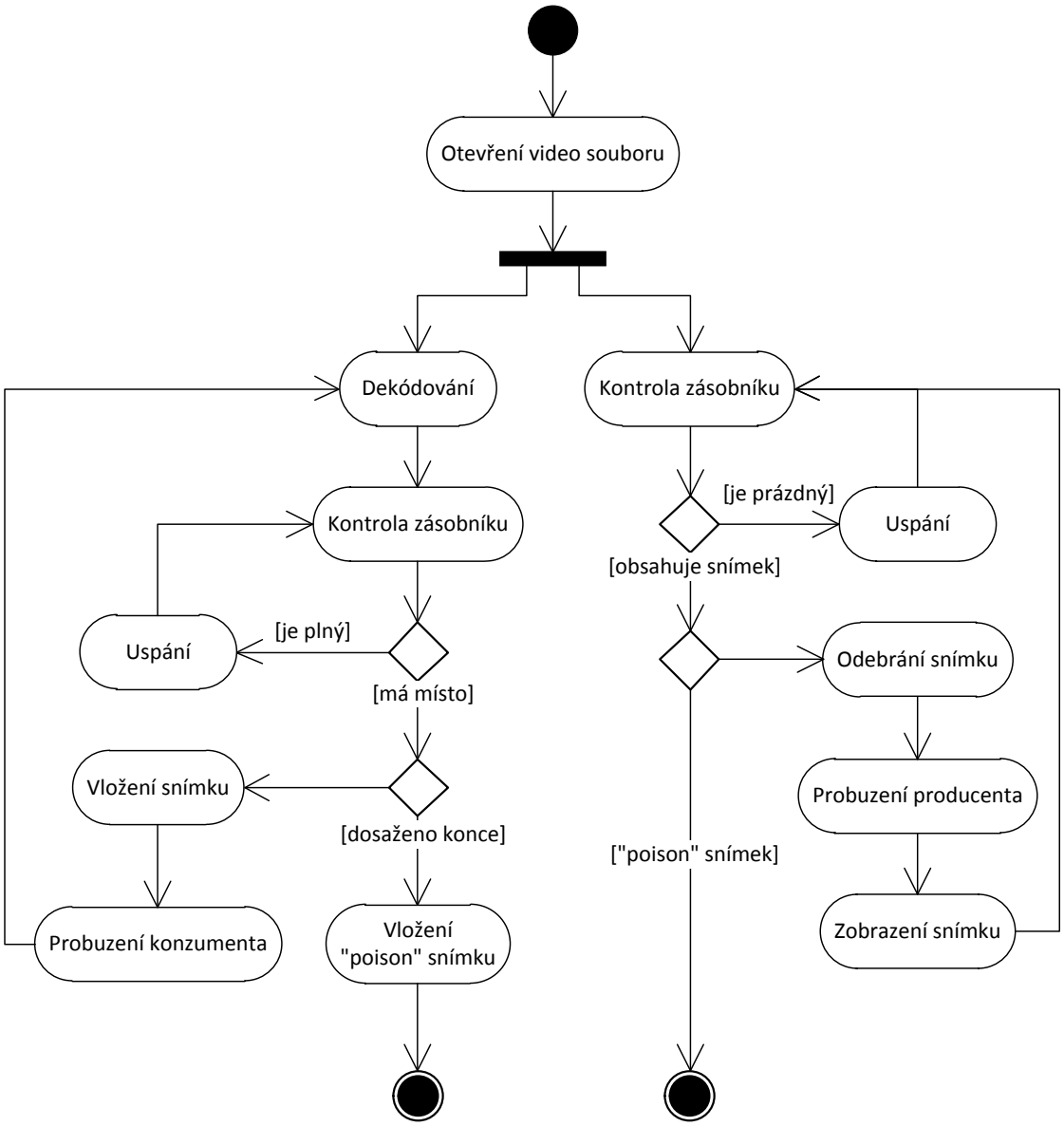
Dále bylo nutné navrhnout video přehrávač, který bude umět přehrát co největší množství formátů videa, a bude podporovat základní funkce, jako jsou Přehrát, Pauza, Stop, Následující snímek, Předchozí snímek a Přechod na určitou pozici. Přehrávač by také měl být schopen plynule přehrávat video, i když se vyskytne nečekaná zátěž systému a snímky se nestíhají dekódovat tak rychle, jak se očekává.

5.1.2 Návrh

Pro minimalizaci „sekání“ videa jsem navrhnul systém producenta a konzumenta, kde producent „vyrábí“ dekódované snímky a ukládá je do zásobníku, zatímco konzument tyto snímky ve správný čas ze zásobníku vytáhne a zobrazí na obrazovku. Toto se děje ve dvou souběžně běžících vláknech. Proto bylo nutné se vypořádat se správnou synchronizací těchto vláken tak, aby se vlákno producenta uspalo vždy, když je zásobník plný a probudilo pokaždé, když je možné do zásobníku opět vkládat. A naopak je stejně tak nutné, aby se konzument uspal vždy, když je zásobník prázdný a probudil, když se do něj něco vloží. Návrh více vysvětluje aktivitní diagram (Obrázek 4).

5.1.3 Implementace

Pro programátory, kteří pracují s více vlákny, nabízí Java celou kolekci tříd (balíček java.util.concurrent), které jsou přímo napsány tak, aby měl programátor co nejméně práce se synchronizací a aby nemusel řešit nutné zamykání objektů. Z této kolekce mě zaujalo především rozhraní BlockingQueue. Třídy implementující toto rozhraní musí kromě základních metod, na které jsme zvyklí u klasických front, implementovat i metody, které při výběru čekají na to, až se vloží nějaký prvek a při vkládání čekají na volné místo. Samozřejmostí je, aby vlákno při čekání nespotřebovávalo výkon procesoru. Jelikož implementace tohoto rozhraní nesmí obsahovat null prvky, navrhují autoři také řešení, jak nejlépe sdělit konzumentovi, že už nebudou vkládány žádné další prvky. Řešením je vložení tzv. „poison“ objektu, který se postará o to, aby se konzument korektně ukončil a nezůstal v nekonečném očekávání dalšího produktu. Pro kon-



Obrázek 4: Aktivitní diagram procesu dekódování

krétní použití jsem zvolil třídu ArrayBlockingQueue, která ukládá objekty v požadovaném pořadí FIFO.

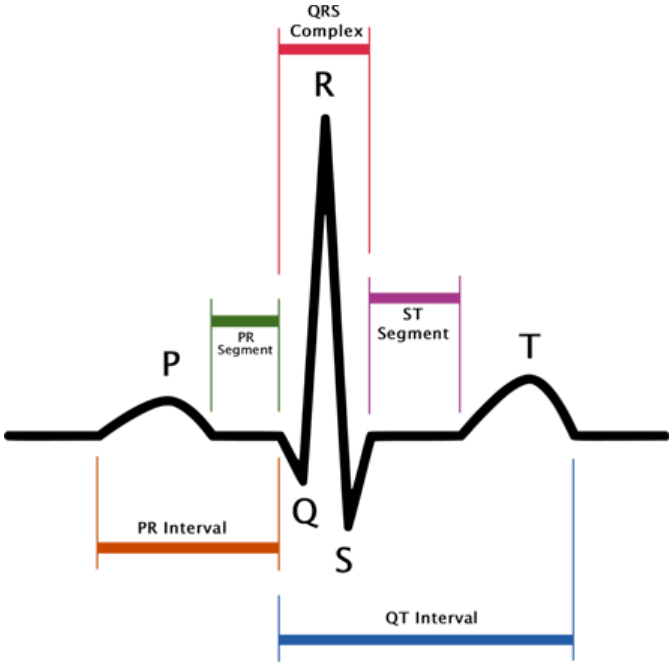
Při přehrávání je také nutné hlásit každou změnu pozice ve videu grafickému rozhraní, aby se mohlo adekvátně obnovit textové políčko s informacemi o aktuální časové známce a číslem aktuálního snímku a také aby se aktualizovala pozice posuvníku. Tento systém je založen na principu událostí a posluchačů.

5.2 Analýza videa

Nejprve bych rád definoval pár základních pojmů.

Elektrokardiografie (EKG) Technika záznamu elektrické činnosti srdce. Elektrody spojené se zaznamenávajícím přístrojem (elektrokardiograf) se umístí na kůži čtyř končetin a na hrudní stěnu. Záznam se nazývá elektrokardiogram [6].

Elektrokardiogram Záznam elektrické aktivity srdce na pohybujícím se papírovém proužku. Tento záznam je pořízen přístrojem nazývaným elektrokardiograf. Užívá se při diagnostice srdečních onemocnění, které mohou způsobit typické změny EKG [6].



Obrázek 5: Běžný průběh signálu EKG[1]

Impuls pro kontrakci myokardu vzniká v tzv. sinoatriálním (SA) uzlu v oblasti pravé předsíně, odkud se šíří dál. Pro účel našeho stručného výkladu je důležité si uvědomit, že tento primární signál je natolik slabý, že jej při běžném záznamu EKG prakticky nezaznamenáme. První vlna signálu, kterou můžeme na EKG záznamu vidět, je vlna P, která svědčí o depolarizaci předsíní, tedy o jejich počínající kontrakci. Samotnou repolarizaci předsíní na EKG nejsme schopni rozpoznat, neboť příslušný biosignál je zastíněn daleko vyšším signálem, pocházejícím od depola-

[1] Zdroj [5]

rizace komor; tento signál je charakterizován komplexem vln QRS. Následující vlna T svědčí o další repolarizaci komor [5].

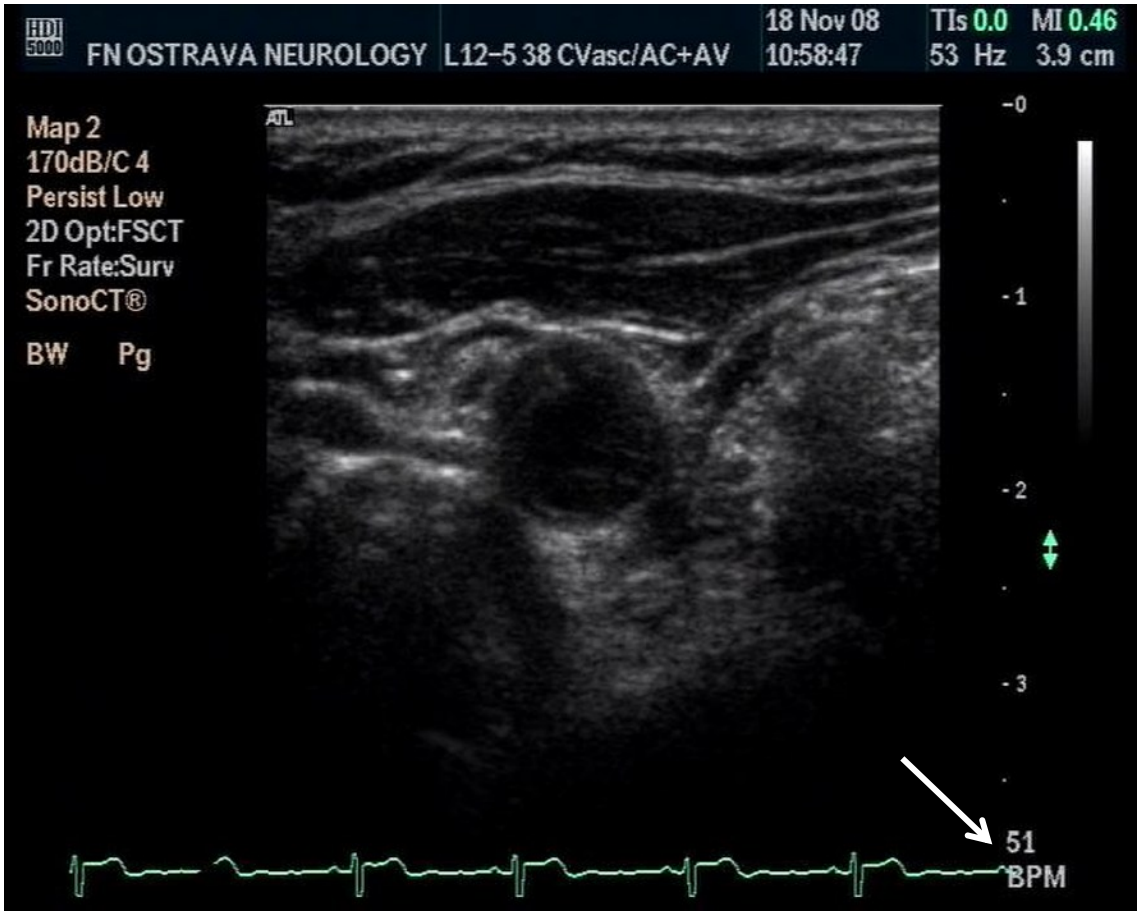
5.2.1 Specifikace požadavků

Dalším krokem je vymyslet způsob, jakým se bude analyzovat video, aby bylo dosaženo požadovaných výsledků. Obrázek 6 nám ukazuje jeden zachycený snímek z analyzovaného videa. Dole na snímku si prosím povšimněte EKG křivky s mezerou uprostřed. Tato mezera znamená místo, které se momentálně překresluje na novou hodnotu. Nejaktuálnější hodnota je v místě, kde končí levá část grafu. Hned za mezerou v oblasti, kde začíná pravá strana grafu, je zaznamenána nejstarší hodnota, která se na dalším snímku překreslí. V tomto grafu je třeba rozpoznat jednotlivá místa tak, že když si uživatel vybere, že ho zajímá, jak vypadá ultrazvukový snímek ve chvíli, kdy nastane např. vlna T, měly by se mu ukázat všechny snímky, které právě probíhající vlnu T zobrazují.

5.2.2 Návrh

Jelikož strojové rozpoznávání oblasti EKG grafu by bylo velice náročné a nespolehlivé, nechám tuto činnost na uživateli. Uživatel bude mít tedy k dispozici nástroj *SelectionTool*, kterým označí oblast, ve které se graf nachází a jen s touto oblastí se pak dále bude pracovat. Bude se jednat o klasický nástroj obdélníkového výběru oblasti, se kterým se již uživatel určitě setkal v mnoha jiných programech. Výhodou práce s menší oblastí je hlavně rychlosti zpracování.

Skutečnost, že se graf neustále dokola překresluje, velmi komplikuje jeho analýzu. Proto by bylo vhodné sestavit kompletní kontinuální pás grafu, se kterým se bude dále lépe pracovat. Nyní si prosím všimněte bílé šipky v pravém dolním rohu. Tady bych rád upozornil na další úskalí, se kterým bylo nutné se vypořádat. Kousek pravé strany grafu je totiž zakrytý písmenkem B a rozpoznávání tvaru grafu je v tomto místě prakticky nemožné. Lidské oko si dokáže chybějící část domyslet, ale pro počítač je to velký problém. O tomto místě se ještě zmíním v následujících odstavcích, kde popíšu návrh zpracování grafu.



Obrázek 6: Ultrazvukový snímek z videa

Sestavení pásu grafu Při sestavování pásu grafu budu procházet každý snímek videa, tedy jen jeho výřez s grafem, který určí uživatel označením oblasti. Na každém tomto snímku je nutné najít, kde se nachází právě vykreslovaná hodnota. Budu proto hledat mezeru, která graf překresluje. V prvním snímku musím tedy projít celý výřez snímku a v každém sloupci hledat, jestli se tam nachází již vykreslený graf. Pokud toto místo najdu, posunu se na další sloupec a hledání opakuji. Pokud ovšem ve sloupci nenaleznu čáru grafu, znamená to, že se může jednat o překreslovací mezeru, ale také se může jednat jen o chybu. Před hledáním je vhodné totiž obrázek předzpracovat prahováním (thresholding). Prahování je funkce, která obecně upravuje hodnoty vstupu podle předpisu [13]:

$$f(c) = \begin{cases} A\ pokud\ c < práh \\ B\ pokud\ c \geq práh \end{cases} \tag{5.1}$$

kde:

- c je vstupní hodnota jasu nebo barvy
- $f(c)$ je výsledná hodnota
- $práh$ je prahovací hodnota
- A, B jsou nové hodnoty pro vstupní hodnotu c pod a nad prahem

V tomto případě bude hodnota A nastavena na 0 a hodnota B na 255, tedy černá a bílá. Hodnotu prahu jsem po dlouhém zkoumání nejlepší hodnoty nastavil na hodnotu 72. Toto předzpracování má za úkol eliminovat šum na pozadí a zvýraznit samotný graf (Obrázek 7). Jako vedlejší efekt ovšem mohou vzniknout i umělé díry a nekonzistence grafu. Proto je třeba nastavit jistou toleranci při hledání vykreslovací mezery, aby se co nejvíce eliminovalo toto riziko.

Obrázek 7: EKG graf předzpracovaný prahováním

Jakmile se nalezne vykreslovací mezera, algoritmus si zapamatuje několik sloupců vlevo od této mezery a pokračuje v hledání na dalším snímku. Nakonec vše spojí dohromady a vznikne nám tak dlouhý pás s odpovídajícím grafem. Na začátku této podkapitoly jsem se zmínil, že nám práci velice komplikuje ono písmeno B, které zakrývá část grafu, a proto je nemožné nalézt v tomto místě vykreslovací mezeru, pokud by se tam nacházela. Jestliže algoritmus nenalezne v celém výřezu žádnou vykreslovací mezeru, má se za to, že je mezera schována za tímto písmenem a tudíž se do paměti uloží jen prázdné místo. Ve výsledku pak má pás s grafem pár mezer, které můžou dělat problémy, pokud v tomto místě měl být vykreslen QRS komplex (viz. Obrázek 5). V následující části budu na pásu grafu vyhledávat jednotlivé QRS komplexy, které odpovídají periodám a na jejichž základě se měří srdeční tep.

Detekce QRS komplexu Na detekci QRS komplexů existuje mnoho algoritmů. Všechny ale počítají s tím, že je vstupem přesný signál a ne nekvalitní obrázek grafu ve špatném rozlišení. Proto jsem navrhnul vlastní algoritmus, jak detekovat QRS komplexy z obrázku. Je založen na výpočtu výběrového rozptylu. Je zřejmé, že pro rovné části grafu bude rozptyl velmi malý, zatímco u výkmitů bude rozptyl mnohokrát větší.

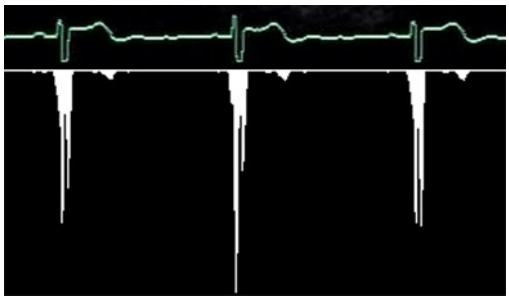
Ve statistice se výběrový rozptyl definuje jako podíl součtu kvadrátů odchylek jednotlivých hodnot od průměru a rozsahu souboru sníženého o jedničku [4], tedy:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}, \tag{5.2}$$

kde:

- x_i je v našem případě výška grafu ve sloupci i, neboli x-ová souřadnice pro $y = i$
- $\bar{x}$ je aritmetický průměr
- n je počet sloupců

Výpočet rozptylu provedu vždy jen pro malou část grafu, řekněme okénko o délce 5px, které při dalším průchodu posunu o 1px doprava. Pro každé takovéto okénko získám číslo, které vyjadřuje rozptyl grafu v tomto okénku. Obrázek 8 ukazuje, jak vypadá sloupcový graf vypočtených hodnot rozptylu pro malou část vstupního EKG grafu. V tomto okamžiku mám tedy množinu rozptylů a mě bude zajímat vždy pouze první největší rozptyl. Pořadové číslo tohoto rozptylu nám odhalí místo QRS komplexu.



Obrázek 8: Výpočet rozptylu

Jakmile naleznu všechny QRS komplexy, mohu si lehce rozdělit graf na jednotlivé periody. Když si uživatel určí, že ho zajímá nějaké konkrétní místo na grafu, přepočítá se toto místo relativně na polohu v té konkrétní periodě a tato relativní poloha se zaznačí i do ostatních period. Tímto způsobem ošetříme to, že jednotlivé periody nejsou stejně dlouhé. Např. při zvýšené tělesné námaze se nám hned zrychlí srdeční tep, což se projeví zkrácením period.

Nyní máme v grafu zaznačené místa, které uživatele zajímají, a už zbývá jen jejich přemapování na čísla snímků, které se mohou uložit na disk jako obrázky.

5.2.3 Implementace

Aby si mohl uživatel označit oblast, ve které se nachází EKG graf, vytvořil jsem nástroj zvaný SelectionTool. Nástroj využívá API Fotomu (Obrázek 9) a lze s ním tedy pracovat jako s kterýmkoliv jiným již implementovaným nástrojem. Při psaní nástroje jsem dbal na to, aby bylo efektivně vyřešeno jeho vykreslování – překresluje se jen ta část plátna, která se změnila. Pro malá plátna je celkem jedno kolik se toho překresluje, ale na velkých plátnech je to již znatelné, např. u akce přesouvání výběru, kdy je třeba překreslovat plátno s nástrojem real-time. Výběr je možné tažením myši přesouvat a měnit velikost. Při implementaci nástroje jsem počítal i s tím, aby uživatel nemohl přetáhnout nebo zvětšit výběr mimo plátno. Nástroj má také množství vlastností, které může uživatel jednoduše měnit v nastavení. Platforma NetBeans nabízí k tomuto účelu okénko, do kterého se vloží nastavitelné vlastnosti.

Integrace vytvořeného nástroje do aplikace spočívá ve vložení tlačítka do palety nástrojů, kterou vytváří *VideoAnalyzerTopComponent*. Nejprve je třeba vytvořit popisný XML soubor, který obsahuje informace o tom, která třída reprezentuje nástroj, dále jaké ikony bude nástroj používat a také jeho název a tooltip.

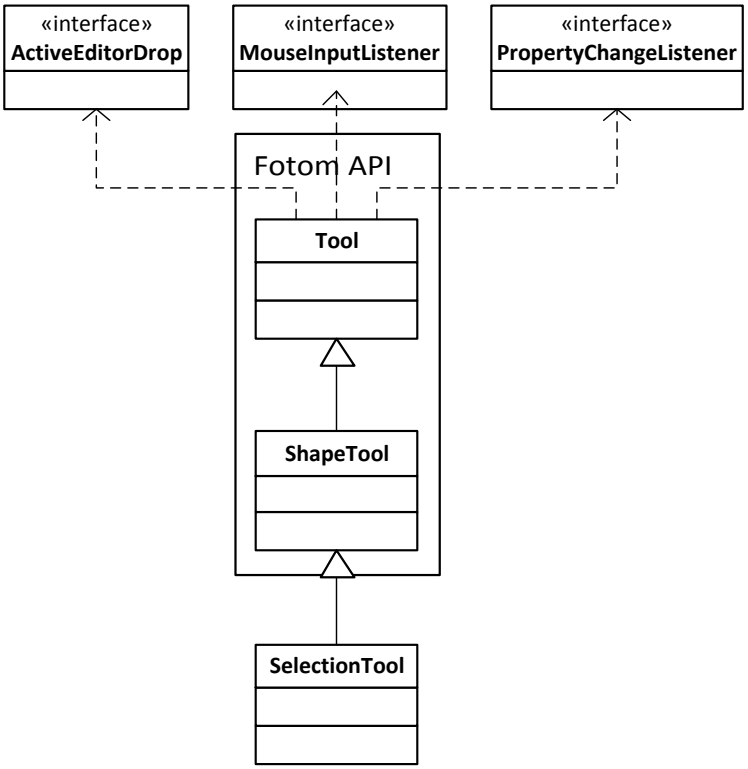
```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE editor_palette_item PUBLIC "-//NetBeans//Editor Palette Item 1.1//EN"
  "http://www.netbeans.org/dtds/editor-palette-item-1_1.dtd">

<editor_palette_item version='1.0'>

    <class name="org.fotomapp.videoanalyzer.SelectionTool" />

    <icon16 urlvalue="org/fotomapp/videoanalyzer/resources/SelectionTool16.png" />
    <icon32 urlvalue="org/fotomapp/videoanalyzer/resources/SelectionTool.png" />

    <description localizing-bundle="org.fotomapp.videoanalyzer.Bundle"
               display-name-key="videoplayer.tools.selection.name"
               tooltip-key="videoplayer.tools.selection.tooltip" />

</editor_palette_item>
```

Výpis 5: Definice nástroje

Definovaný nástroj je ještě třeba vložit do samotné palety, která nám ho zobrazí jako tlačítko s definovanou ikonou a názvem. Vkládání se provádí v souboru *layer.xml* (Výpis 6), kde *VideoAnalyzerToolPalette* je název palety, do které se nástroj vloží, což umožňuje, aby měl každý modul svou vlastní paletu, Tools je název kategorie a *SelectionTool.xml* je definice nástroje (Výpis 5).

```
<folder name="VideoAnalyzerToolPalette">
   <folder name="Tools">
      <file name="SelectionTool.xml" url="resources/SelectionTool.xml" />
    </folder>
</folder>
```

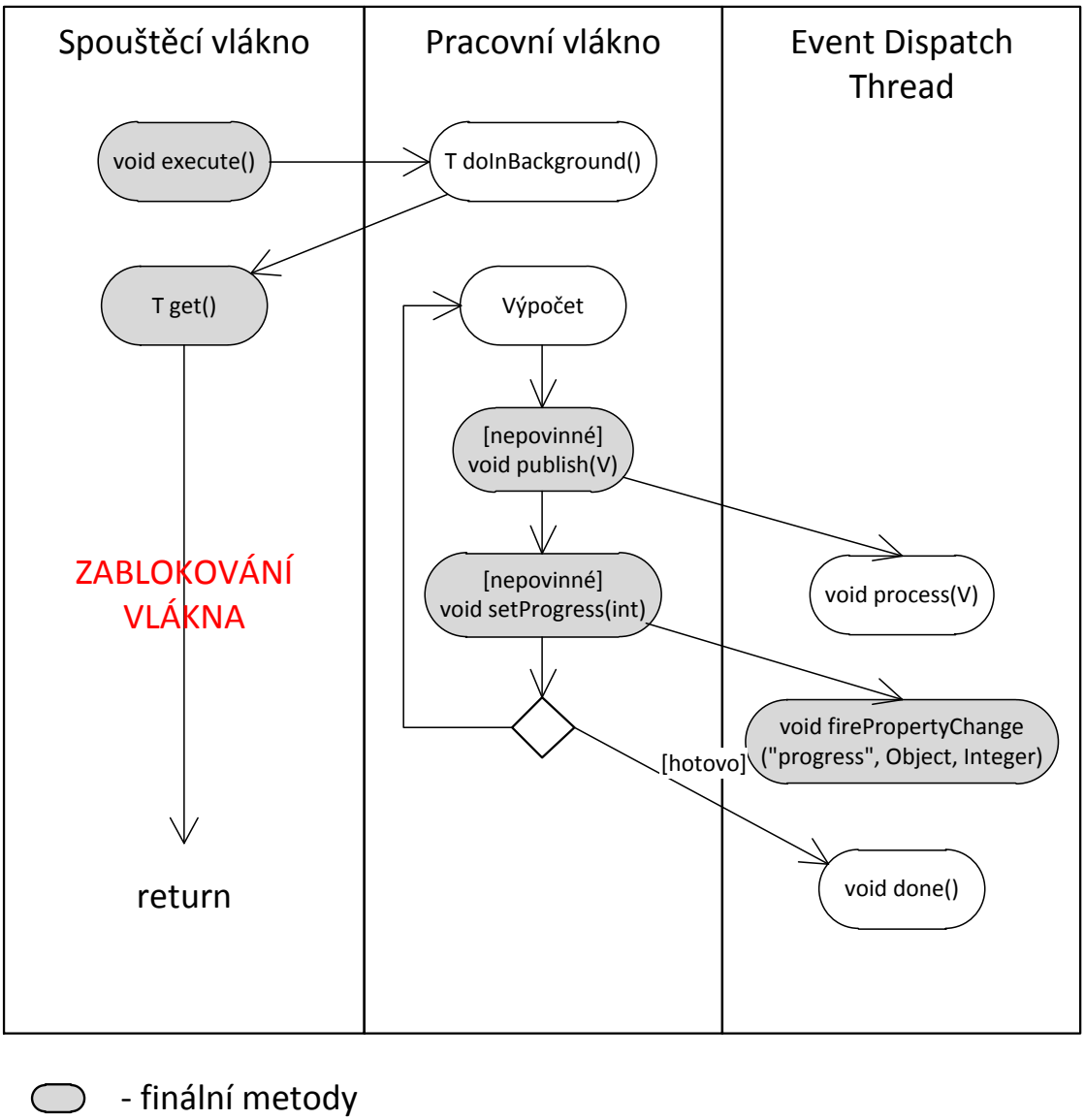
Výpis 6: Vložení nástroje do palety



Obrázek 9: Třídní diagram nástroje SelectionTool

Sestavení pásu grafu Pro sestavování grafu jsem připravil hlavní třídu *EkgBuilder*. Tato třída dědí z třídy *SwingWorker* a implementuje rozhraní *Cancellable*. To nám říká, že se může spouštět na pozadí a že se také může její běh zastavit. Běh dlouho trvajících úkolů na pozadí je v grafických aplikacích velice důležitý, protože když tento úkol běží ve svém vlastním vlákně, nezablokuje se tak GUI, které se může dále překreslovat na základě uživatelských podnětů. Použití více vláken v jedné aplikaci ovšem nepřináší jen výhody, ale také problémy s jejich synchronizací. Programátor se musí postarat o to, aby dvě souběžné vlákna nevstupovaly zároveň do stejné kritické sekce. Toho se v jazyce Java docílí použitím klíčového slova *synchronized* a označením bloku, který se má synchronizovat. Dále by se měly zobrazovat grafické prvky aplikace a provádět přístup ke GUI pouze z EDT. EDT je vlákno, jehož hlavním úkolem je

zpracovávat události z AWT nebo Swingu. Pokud již nutně potřebujeme aktualizovat GUI z nějakého jiného vlákna než EDT, docílíme toho pomocí metody *SwingUtilities.InvokeLater(Runnable)*. Třída *SwingWorker* (Obrázek 10) již ovšem s tímto přístupem počítá, takže pro práci na pozadí použije programátor metodu *doInBackground()*. Ta běží sama ve vlastním vlákně a když skončí svůj běh, předá automaticky řízení metodě *done()*, která je opět spuštěna v EDT, takže se výsledek práce může lehce zobrazit uživateli.



Obrázek 10: Grafické znázornění SwingWorkeru

V průběhu sestavování EKG grafu i během jeho analýzy pracuji s obrazem, který prošel předzpracováním prahování. K tomuto účelu mi posloužily třídy z jádra aplikace *RgbToGrayscale* a *BinarizeTreshold*. První slouží k převodu obrázku do stupňů šedi a druhá provede samotné prahování.

Detekce QRS komplexu K výpočtu rozptylu, který se v detekci QRS komplexu používá, slouží třída Variance. Při vytváření instance se předá konstruktoru malé okénko grafu, ze kterého se rozptyl vypočte a metodou getVariance() se vrátí jeho hodnota. Grafickou reprezentaci vypočtených hodnot pro malý výřez grafu ukazuje Obrázek 8. Na tomto obrázku si můžete všimnout, že výpočtem vzniknou pro každou periodu dvě lokální maxima. První označuje hledaný QRS komplex a druhé maximum odpovídá umístění vlny S (viz. Obrázek 5). Jakmile mám vypočteny všechny rozptyly, začnu je popořadě procházet a porovnávat. Pokud je rozptyl větší než 27, má se za to, že se jedná o QRS komplex a jeho poloha se zaznamená. Po nalezení jednoho QRS komplexu přeskočím o pár pixelů doprava, abych se vyhnul falešné detekci, protože vlna S má hodnoty rozptylu také větší než 27. Takto najdu všechny QRS komplexy a podle jejich umístění rozdělím graf na jednotlivé intervaly. Po kliknutí myši do oblasti sestaveného grafu se zobrazí náhledy snímků, které odpovídají stejné události v každém intervalu. Tyto snímky je možné kdykoliv uložit na disk. Více o ukládání snímků v kapitole 5.3.

Nalezené hranice interval ovšem nejsou konečné. Pokud není uživatel spokojený s výsledkem automatické detekce, má ještě možnost ručního nastavení hranic intervalů a to tak, že je jednoduše přetáhne na nové místo. Při tažení první hranice intervalu dochází k posunu všech hranic najednou a při tažení jakékoliv jiné hranice dochází ke změně měřítka. I po nastavení nových hranic je možnost se vrátit zpět na původní hranice, protože se staré hranice pořád udržují v paměti.

5.3 Ukládání snímků

5.3.1 Specifikace požadavků

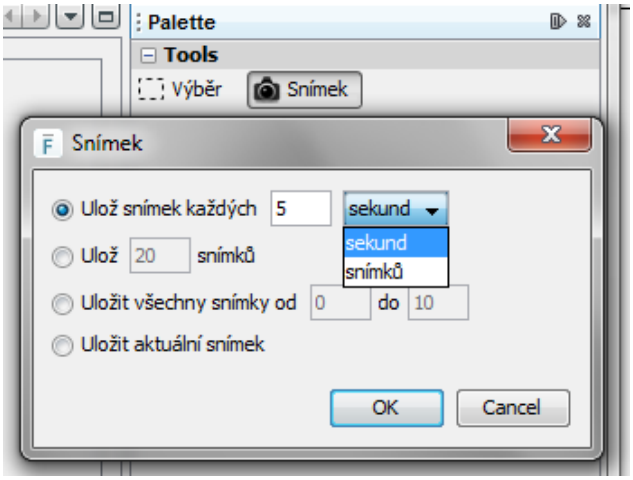
Posledním požadavkem je možnost ukládat vybrané snímky na disk nepravidelně na základě událostí v EKG grafu a také podle pravidel, které si bude moci uživatel vybrat a sám nastavit. Prvním pravidlem bude možnost vytáhnout z celého videa snímek každých x sekund nebo každých x snímků. Dále by měl mít uživatel možnost uložit z videa např. 20 snímků, které jsou z videa vybrány v pravidelných intervalech. Třetím pravidlem je možnost uložit všechny snímky, které jsou v určitém intervalu a poslední možnost je uložit aktuálně zobrazený snímek.

5.3.2 Návrh

Pro univerzální využití navrhnu metodu, která vrátí obrázek na pozici, která jí byla předána parametrem. Pro výběr možností, které snímky se uloží, bude mít uživatel k dispozici panel s tlačítky a textovými poli, kde si vše potřebné sám nastaví. O samotné ukládání snímků se pak bude starat třída k tomu určená a bude umět ukládat snímky v základních formátech BMP, JPEG, GIF a PNG.

5.3.3 Implementace

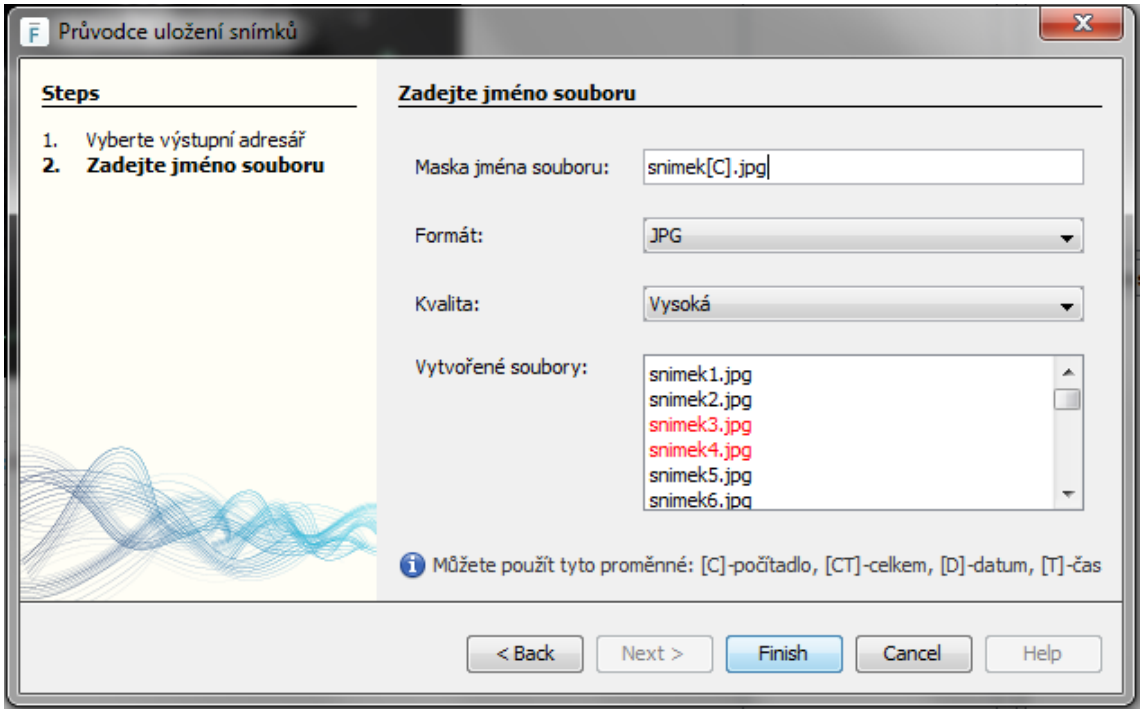
Pro jednoduché nastavení jsem vytvořil v paletě nástroj, který zobrazí panel se zaškrtávacími tlačítky a textovými poli, kde si uživatel nastaví, jaké snímky chce uložit (viz. Obrázek 11). Při ukládání snímků se zobrazí průvodce uložení snímků (Obrázek 12), kde je možnost zvolit výstupní adresář, kam se snímky uloží, masku jejich názvu, formát a pokud se jedná o jpeg, tak i kvalitu. Před samotným uložením je k dispozici náhled souborů, které budou vytvořeny a pokud již některé soubory existují, zvýrazní se v tomto seznamu červeně. Při ukládání se pak ještě program pro jistotu ptá, jestli se má existující soubor opravdu přepsat.



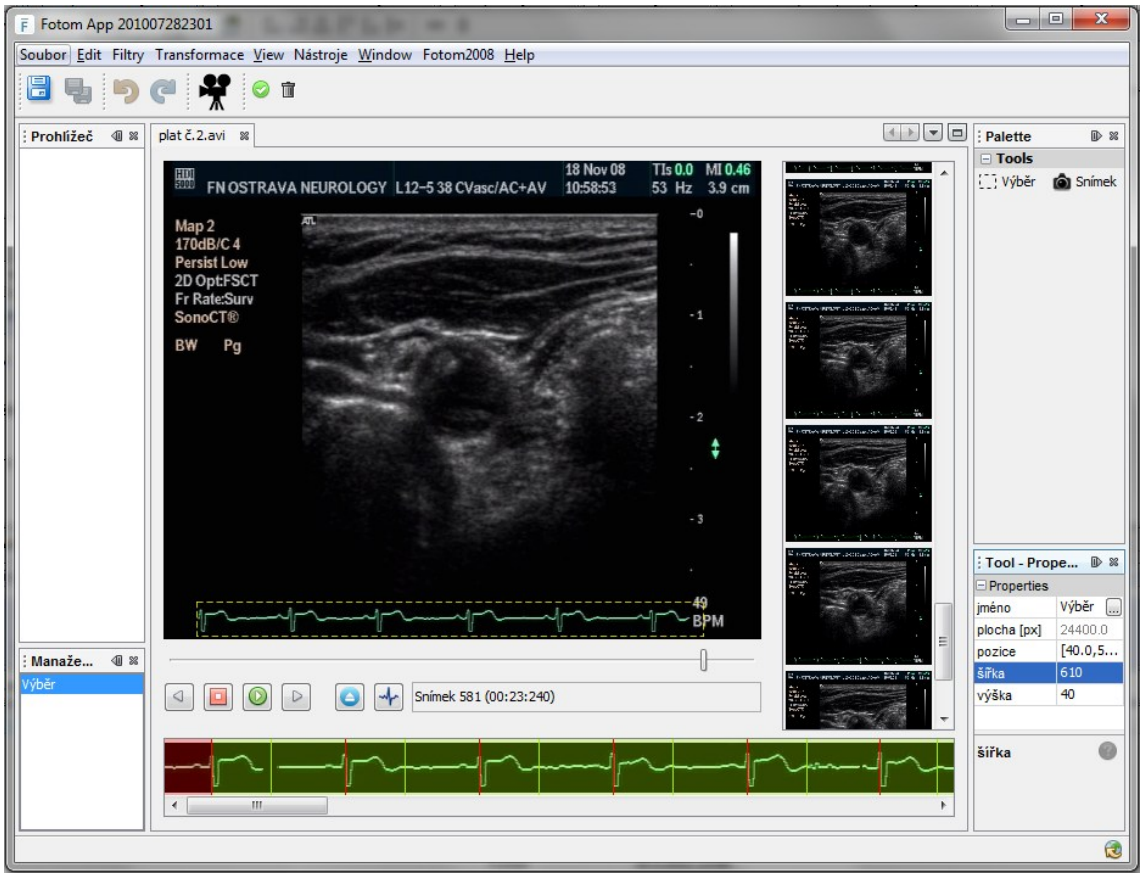
Obrázek 11: Možnosti nástroje Snímek

Obrázek 13 demonstruje vzhled výsledného modulu. V levé horní části je okno prohlížeče, ve kterém se zobrazují uložené FTM objekty. Levá spodní část obsahuje seznam objektů, definovaných na plátně. V tomto případě lze na plátno přehrávače přidávat pouze objekty nástroje výběru, takže toto je jediný typ, který se v seznamu objeví. Definované objekty lze upravovat a dle libosti mazat. V pravé části se nachází paleta s nástroji a pod ní okno, kde se dá přesněji nastavit velikost výběru. Zbývá hlavní okno aplikace, kde se přehrává video a vykresluje graf spolu s vybranými snímky. Přehrávač lze ovládat pomocí tlačítek umístěných pod jeho plátnem

a video je možné také točením kolečka myši přibližovat a oddalovat. Vedle ovládacích prvků jsem umístil informační proužek, ve kterém se vypisuje číslo aktuálního snímku a jeho čas. Pod tlačítky je vymezen prostor pro vykreslení sestaveného pásu grafu. Vykreslený graf umožňuje a očekává interakci uživatele. Ten si tak může vybrat místo, které ho na grafu zajímá a po výpočtu všech ostatních odpovídajících míst se napravo od plátna přehrávače zobrazí náhledy vybraných snímků.



Obrázek 12: Průvodce uložení snímků



Obrázek 13: Celkový pohled na výsledný modul

6 Závěr

Cílem mé diplomové práce bylo vyvinout modul pro provedení „rozsekání“ video záznamů na jednotlivé snímky různých formátů dle událostí zachycených na tomto záznamu. Dle mého názoru byl cíl práce splněn a výsledný modul byl poté integrován do systému FOTOM 2009. V současné době jsou do systému FOTOM 2009 vyvíjeny další dva moduly. První vyvíjí Bc. Tomáš Pytlík a jeho modul umožní 2D modelování. Autorem druhého modulu je Tomáš Hudeček a jeho modul poskytuje rozšířené možnosti prahování, definici objektů pomocí Bézierovy křivky a filtry pro úpravu jasu a kontrastu. Doplnění stávající verze Fotomu o tyto tři nové moduly vytvoří novou verzi aplikace – FOTOM NG.

Při psaní diplomové práce jsem se seznámil s problematikou počítačového zpracování fotografií za účelem následného numerického zpracování a s vědním oborem zvaným fotogrammetrie. Jedná se o vědní obor sloužící v medicíně k vyšetření tkání neinvazivní metodou. Jako snímač se nejčastěji používá ultrazvuková sonda, která pracuje na principu odrazu ultrazvuku od tkání s různou hustotou.

Dále jsem se seznámil s možnostmi systému FOTOM 2009 a naučil se efektivně pracovat s jeho jádrem. Systém FOTOM 2009 nabízí velice propracované API, které poskytuje snadný způsob komunikace mezi moduly. Toho jsem maximálně využil i při tvorbě svého modulu. Také jsem se naučil vyvíjet aplikace na platformě NetBeans, na které byl tento systém postaven.

Ve výsledku jsem vyvinul robustní modul, který poskytuje možnost načíst a přehrát videa různých formátů. Uživateli se tak dostává k dispozici mocný nástroj určený hlavně k ulehčení analýzy ultrazvukových záznamů, ale věřím, že se pro něj najde využití i mimo lékařské prostředí. Modul poskytuje základní funkce video přehrávače, umí z videa uložit jednotlivé snímky na základě uživatelského nastavení a také analyzovat video a rozpoznávat tak jednotlivé periody EKG grafu, který je na video záznamu zachycen. Uživatel má také možnost nastavit si sám intervaly, pokud nebyly rozpoznány nebo pokud byly rozpoznány špatně. Při tvorbě video přehrávače jsem kladl důraz na plynulost přehrávání i při neočekávané zátěži systému.

Pomocí tohoto modulu lze analyzovat videozáznam ultrazvukového vyšetření a tím umožnit lepší prevenci vzniku cévní mozkové příhody (CMP). CMP je v současnosti třetí nejčastější příčinou úmrtí v lidské populaci. Jedním z řešení, jak snížit výskyt CMP je právě prevence a její včasné odhalení. Vyšetření ultrazvukovou sondou je momentálně nejvhodnější metodou pro toto sledování a pomocí vytvořeného modulu je možno snímky lépe analyzovat.

Do budoucna bych navrhoval reagovat na vznik nových formátů videa a jejich podporou rozšířit tento modul. Dále by bylo vhodné rozšíření modulu o časovou osu, na které si bude moci uživatel jednodušeji nastavit jednotlivé intervaly. Později také mohou vzniknout nové typy video záznamů, které nebudou mít signál, podle kterého se provádí analýza, vykreslený přímo

do videa, ale tento signál může být ve zvláštním souboru uložený např. jako číselné hodnoty. Jakmile tedy vzniknou tyto nové typy záznamů, bude určitě vhodné vyvinout pro ně další modul, který si s nimi bude rozumět.

7 Literatura

[1] BOUDREAU, T. – TULACH, J. – WIELENGA G. *Rich Client Programming: Plugging into the NetBeans Platform*, 1. vyd. Prentice Hall, 2007.640 s. ISBN 978-0-13-235480-6

[2] BÖCK, H. *Platforma NetBeans: Podrobný průvodce programátora*. 1. vyd. Brno: Computer Press, 2010. 320 s. ISBN 978-80-251-3116-9

[3] BÖHM, J. *Fotogrammetrie* [online]. 2002 [cit. 2011-04-13] URL: <http://igdm.vsb.cz/igdm/materialy/Fotogrammetrie.pdf>

[4] BRIŠ, R. – LITSCHMANNOVÁ, M. *Statistika I.: pro kombinované a distanční studium* [online]. c2004. [cit. 2011-04-20] URL: <http://is457.vsb.cz/bris/Teaching/Statistika 1/Kapitola 1.PDF>

[5] *Elektrokardiogram – Wikipedie* [online]. 2011, poslední revize 5. března 2011 [cit. 2011-04-23] URL: < http://cs.wikipedia.org/wiki/Elektrokardiogram>

[6] KOLEKTIV AUTORŮ. *Výkladový ošetřovatelský slovník*. 1. vyd. Praha: Grada, 2007. 568 s. ISBN 978-80-247-2240-5

[7] KRAHULEC, L. *Počítačové zpracování fotografie*. Ostrava, 2009. 57 s. Diplomová práce na Fakultě elektrotechniky a informatiky Vysoké školy báňské – Technické univerzity Ostrava na katedře informatiky. Vedoucí diplomové práce Doc. Ing. Lačezar Ličev, CSc.

[8] LIČEV, L. *Analýza, modelování, rozpoznávání a vizualizace procesu měření objektů na snímcích.* Ed. Computer Press. 1. vyd. Brno: Computer Press, 2010. 125 s. ISBN 978-80-251-3296-8

[9] LIČEV, L. *Systém FOTOM 2007 a vizualizace procesu měření.* Ostrava: Tanger, spol. s r.o., 2008, vol. 2008, GIS2008, GIS Ostrava, s. 49-50, Tanger, Spol. s r. o., ISBN 978-80-254-1340-1

[10] LIČEV, L. *Systém FOTOM 2008 a vizualizace procesu měření*. Ed. Ing. Kateřina Pešková, Ostrava: Tanger, spol. s r. o., 2009, vol. 2009, čis. 1, 35, VŠB-TUO, Hornicko-geologická fakulta, Institut geoinformatiky, ISBN 978-80-87294-00-0

[11] NETBEANS.ORG. *NetBeans API* [online]. 2011 [cit. 2011-04-05] URL: <http://bits.netbeans.org/dev/javadoc/>

[12] *Overview (xuggle-xuggler)* [online]. 2010, poslední revize 8. března 2010 [cit. 2011-04-10] URL: <http://build.xuggle.com/view/Stable/job/xuggler_jdk5_stable/javadoc/java/api>

[13] *Prahování – Wikipedie* [online]. 2011, poslední revize 12. ledna 2011 [cit 2011-04-16] URL: <http://cs.wikipedia.org/wiki/Prahování>

[14] SPENCE, D. J. *Mozková mrtvice*. 1. vyd. Praha: Triton, 2008. 256 s. ISBN 978-80-7387-058-4

[15] SUN Microsystems. *Programming in Java Advanced Imaging* [online]. 1999, poslední revize 18. prosince 2007 [cit. 2011-04-13] URL: <http://java.sun.com/products/java-media/jai/forDevelopers/jai1_0_1guide-unc/>

A Přílohy na CD

- Zdrojové kódy
- Spustitelná distribuce programu
- Uživatelská příručka
- Programátorská příručka